Distributed Boundary Coverage with a Team of Networked Miniature Robots using a Robust Market-Based Algorithm

Patrick Amstutz[1], Nikolaus Correll[2], and Alcherio Martinoli[1]

[1]Distributed Intelligent Systems and Algorithms Laboratory
École Polytechnique Fédérale Lausanne, CH-1015 Lausanne, Switzerland

[2]Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, 02139 Cambridge, MA, USA

March 13, 2009

Abstract

We study distributed boundary coverage of known environments using a team of miniature robots. Distributed boundary coverage is an instance of the multi-robot task-allocation problem and has applications in inspection, cleaning, and painting among others. The proposed algorithm is robust to sensor and actuator noise, failure of individual robots, and communication loss. We use a market-based algorithm with known lower bounds on the performance to allocate the environmental objects of interest among the team of robots. The coverage time for systems subject to sensor and actuator noise is significantly shortened by on-line task re-allocation. The complexity and convergence properties of the algorithm are formally analyzed. The system performance is systematically analyzed at two different microscopic modeling levels, using agent-based, discrete-event and module-based, realistic simulators. Finally, results obtained in simulation are validated using a team of Alice miniature robots involved in a distributed inspection case study.

Keywords

boundary coverage, distributed coverage, market-based algorithms, multi-robot systems, miniature robots, networked robots

Mathematics Subject Classifcations (2000)

68T37 Reasoning under uncertainty, 68T40 Robotics, 68W15 Distributed algorithms, 93C65 Discrete event systems

*This work has been done when all authors were with the Swarm-Intelligent Systems Group, EPFL

1 Introduction

We wish to develop distributed control algorithms and formally analyze their performance on distributed systems composed of miniature robots, which are extremely unreliable and subject to sensor and actuator noise. This paper presents an algorithm, its analysis and experimental validation for the multi-robot boundary coverage problem. Multi-robot coverage [17, 18, 29] requires the coordination of a team of robots to entirely cover an environment with a specific end-effector (sensor or actuator) mounted on each robot. Coverage has a variety of industrial, humanitarian, and military applications such as inspection [7], cleaning [18], painting, plowing, mowing, and de-mining [1]. In the area coverage problem, robots are required to cover an entire area using one of their sensors or end-effectors. Boundary coverage is an instance of the area coverage problem that requires the coverage of the boundaries of all objects in the environment, and is equivalent to cover all the area that is close to the boundaries. The distributed versions of both coverage problems consist in distributing the whole area to be covered among the robots while minimizing a user-defined objective function. Example of objective functions are the area allocated to the robot that covers the most area, overall task execution time, or the aggregated difference among all robots, to name a few.

Robotic agents embedded in the real world are subject to sensor and actuator noise and cannot be assumed to reliably perform the desired actions within desired time bounds. This is in particular the case for miniature robots such as those considered in this paper. Moreover, communication among the team of robots is prone to packet loss or even disconnection, e.g. due to limited range. One significant source of actuator noise is *wheel slip* which in turn implies navigation errors during the mission execution and non-deterministic delays for moving from one waypoint to another. This property in particular renders off-line planning, i.e. planning all trajectories of the robots beforehand, as an exclusive method for coordination unsuitable for multi-robot systems. Likewise, predicting the performance of a multi-robot system deterministically yields little insight into the actual performance, which is potentially better analyzed probabilistically..

1.1 Contribution of this paper

We provide a comprehensive study of a state-of-the-art deterministic algorithm for multi-robot routing with provable performance guarantees for totally reliable platforms [20] applied to distributed boundary coverage using miniature robots, which are unreliable due to sensor and actuator noise. We show how sensor and actuator noise can be accomodated by continuous replanning and extend the auction algorithm to accommodate dynamic team sizes that arise from unreliable or limited-range communication. For analyzing the performance of our approach, we combine deterministic analysis with a probabilistic modeling framework which allows us for faithfully reproducing sensor and actuator noise of the individual robotic platform. We validate our findings on a team of five miniature robots Alice and a distributed boundary coverage case study.

1.2 Outline

This section is concluded with an overview over related work. The distributed boundary coverage problem and related metrics are introduced in Section 2. We then describe the experimental

setup in more detail (Section 3). The market-based approach is formally described (Section 4), and its complexity in terms of computation and communication is discussed. We then describe the two microscopic models used for analysis of desired properties of the algorithm in Section 6. Experimental and numerical results (Section 7) are followed by a discussion (Section 8) and a conclusion (Section 9).

1.3 Related Work

The multi-robot task allocation problem is a canonical problem in multi-robot systems, with multi-robot coverage being a specific instance. For allocating tasks to a team of robots, various architectures ranging from reactive to deliberative algorithms have been proposed and studied (see [15] and references therein for an overview). More recently, so called market- or auction-based algorithms have been proposed for this purpose [10]. In a market-based algorithm tasks are represented as commodities, which are traded by the robots based on the cost that task execution would impose on each robot. In particular, MURDOCH [16] and *Hoplites* [19], among others, have shown the viability of market-based approaches for task-allocation on real robots. Specifically for multi-robot coverage, [25] applies a market-based algorithm for arbitrating coverage among the robots. Zlot et al. [30] studied complex task-allocation problems using indoor and outdoor real robots. Dias et al. have investigated various problem domains using market-based approaches [9], some of them concerned with multi-robot exploration and mapping [31].

Other solutions to the multi-robot coverage problem, which are provably complete and allow for calculating lower bounds on the performance under ideal conditions are motivated from operation research. For instance, Easton et al. [11] proposes a provably complete, off-line generated, near-optimal algorithm inspired by the *Chinese Postman Problem* [13]. Easton et al.'s algorithm, which is a constructive heuristic, assures coverage of all edges in a graph, where edges represent the boundary of objects in the environment as well as navigable routes between them. Off-line planning limitations were relaxed in a successive contribution, which took into account possible failures of the robotic nodes [28]. In [29], Zheng et al. propose a multi-robot coverage with interesting lower bounds on the performance by subdividing the graph representing the environment in a forest using an algorithm presented in [12]. Although these approaches might yield comparable or better lower-bounds for the performance than a market-based approach, we feel that the latter is best suited for implementation on a distributed robotic system with unreliable communication due to its potential for distributed computation.

When modeling multi-robot coverage, completeness of the allocation as well as upper and lower bounds are commonly the most interesting properties. Analysis is often done by decomposing the coverage algorithm into algorithms with provable completeness. Examples of this are the Spanning Tree Coverage algorithm [14] which is used for multi-robot coverage work presented in [17] and references therein, or the before mentioned tree coverage algorithm used in [29]. More recently Lagoudakis et al. [20] presented upper and lower bounds for the performance of a market-based algorithm when applied to the multi-robot routing problem, which can be casted into a distributed coverage problem and is also the basis for the algorithms developed in this paper. Whereas upper and lower bounds and completness are discrete properties that are shown analytically in all of the above work, the effective performance when executed on a real system, however, is usually evaluated empirically by running the algorithm for various problem instances and with or without simulating unreliable robotic platforms.

For probablistic modeling of distributed systems composed of large numbers of unreliable robots Martinoli et al. [23] and Lerman et al. [21] propose a probabilistic modeling methodology that bases on the master equation for physical systems (see also [22]), and validate their findings with real robots in a distributed collaboration case study [23] and using realistic simulation for a foraging task [21]. In their work, the average number of robots in a specific state and discrete environmental variables are represented by a system of coupled difference or differential equations, respectively. This methodology has also been used for modeling a fully reactive, distributed controller for the same case study considered in this paper [5], but reaches its limitations when robots are deliberately controlled and exchange information [6] as the number of possible system states becomes extremely large and the required state transition probabilities are difficult, if not impossible, to calculate.

2 Distributed Boundary Coverage

We study the multi-robot boundary coverage problem using a case study concerned with covering the boundaries of elements aligned in a repetitive, grid-like pattern. For structures with regularly spaced elements (see Figure 1), boundary coverage is dual to visit and cover all cells of a grid by a team of robots (as it is the case in [14, 17, 29]).

2.1 Case Study

As case study serves the inspection of the boundaries of all blades in a mock-up turbine scenario. The turbine environment consists of 25 blades that are arranged in a 5x5 pattern. We consider all boundaries to be inspected, if each blade has been completely circumnavigated by at least one robot. The mock-up turbine scenario has been implemented both physically and in realistic simulation using *Webots* [24] and is shown in Figure 1, *left* and *right*, respectively. In the remainder of this paper, we consider each boundary that needs to be covered as a task and the distributed boundary coverage problem as a task-allocation problem. The task-allocation problem then consists of finding the distribution of tasks to the robots that minimizes the longest execution time among all robots under consideration of sensor and actuator noise.

As each task has a unique location in the environment, we formulate the task allocation problem on a graph with tasks as vertices and the shortest possible routes between the tasks as edges. A possible task allocation for the case study of this paper is depicted in Figure 2, *left*. Figure 2 also shows (*right*) the case of robot navigation error and a possible task re-allocation.

3 Experimental Setup

This section describes the turbine mock-up environment and the robotic platform (both hardware and software) in more detail.

3.1 Turbine Mock-up Environment and Robotic Platform

A 60cm×65cm arena is populated with 25 blades in a regular pattern (see Figure 1), mimicking the blades of a jet turbine.

The Alice II robot [4] is endowed with a PIC micro controller (368 bytes RAM, 8Kb FLASH), has a length of 22mm, and a maximal speed of 4cm/s (Figure 1, *right*). Four IR modules can

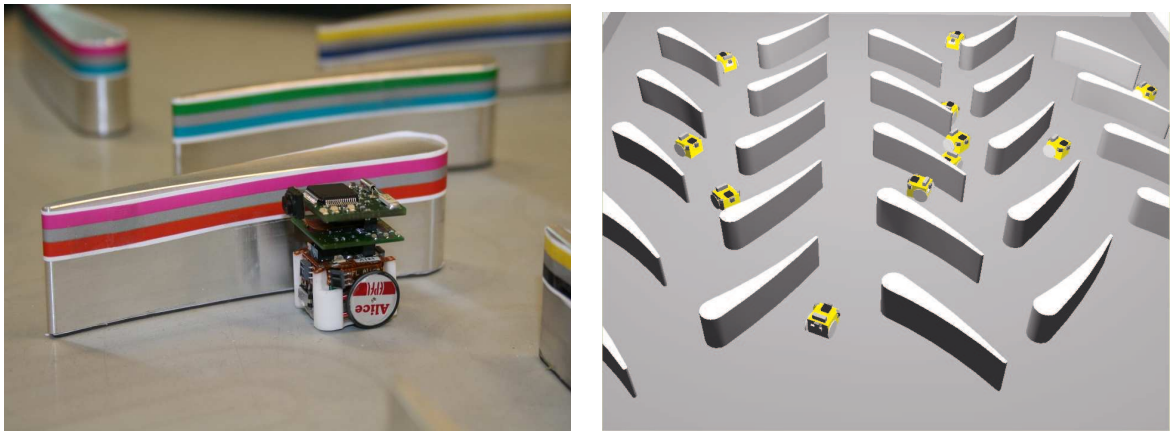

Figure 1: *Left*: Close-up on an Alice robot equipped with radio module and camera in the real setup. Every blade is marked with an unique color code, which enables localization in the environment. *Right*: Overview of the turbine set-up in the realistic simulator. Because of computational efficiency, localization is obtained through an equivalent function (including noise) that is implemented at the level of the simulator supervisor modules.

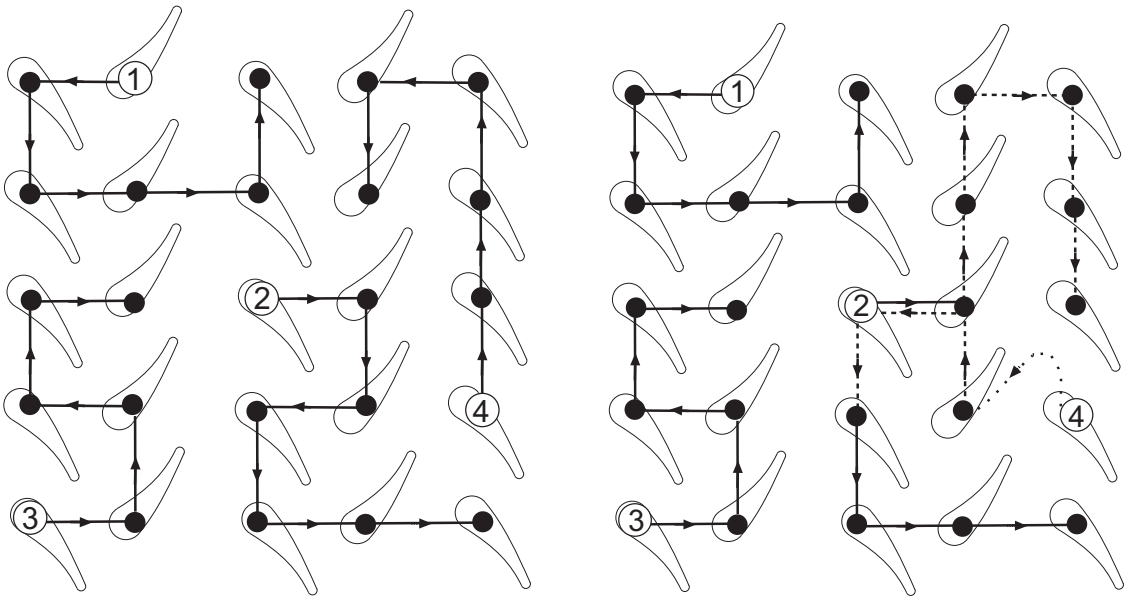

Figure 2: *Left:* Possible trajectories for four robots in a 5x5 blade environment (bold line). *Right:* Robot ④ fails executing its task sequence due to navigation error. Task re-allocation leads to a new allocation for robots ② and ④.

serve as crude proximity sensors (up to 3cm) and local communication devices (up to 6 cm). Robots are able to distinguish between blades, the arena boundaries, and other robots. For inspection and localization, we use a VGA color CCD camera that is sub-sampled to 30x30 pixels. The upper parts of the blades are equipped with unique color markers that consist of three colored horizontal bars. The presence or absence of the 3 color channels (red, green, and blue) is used to encode 3 bits per color. Using the middle gray bar as reference (all channels at 50%) allows us to encode 64 different codes of which we are using 25 to identify each blade [26]. For communication and eventually transmitting sensory information among the robots and to a base station, a 2.4GHz radio module running TinyOS[1] is used. The radio also provides additional computational power and memory (8MHz CPU, 4k RAM, 4MBit flash memory).

As the computational capabilities of the Alice robots are limited, parts of the computation required by the algorithm presented in this paper are executed on an external computer (Intel Pentium IV workstation with a Telos MoteIV mote[2] attached to it), with which robots communicate the current set of inspected blades along with their identification number and receive the task allocated to them as a sequence of blades to inspect. We let the base station continuously broadcast the auction results, in order to enable the individual robots to turn on their radio only briefly (for around 0.1s) and only before starting a new task in order to save energy.

Task progress is monitored using the external computer and an overhead camera that is processed using the open-source tracking software *SwisTrack*[3] [8]. Notice that although the computation of the task-allocation is executed on an external computer due to the limitations of the platform in use, the algorithm developed in this paper could indeed be implemented in a fully distributed fashion exclusively using on-board computational capabilities.

3.2 Robot Controller

Moving from blade to blade (i.e. traversing an edge of the graph representing the environment) is achieved by navigation along specific waypoints on a blade's boundary (see Figure 3), which can be distinguished by the robot's on-board sensors. More specifically, robots can detect a blade's tip as well as measure the curvature of the blade using odometry, and thus roughly estimate where they are located relative to the blade's boundary. After arriving at one of two specific waypoints, a robot uses a pre-programmed (open-loop) sequence of movements to navigate to one of the four neighboring blades.

Although this experiment is very specific, the described implementation is representative for a sensor-based navigation paradigmn that is prone to navigation errors due to sensor noise or wheel slip.

4 Market-Based Task Allocation

In market-based task allocation [9], coordination is achieved by trading tasks using auctions. The value of a task for a particular robot is evaluated based on an individual, or local, objective function. A basic bidding scheme is known as *single-item* auction, in which robots bid on one single item at a time. The robot which submitted the best bid according to a given rule wins

[1] http://www.tinyos.net/
[2] http://www.moteiv.com/
[3] http://swistrack.sourceforge.net

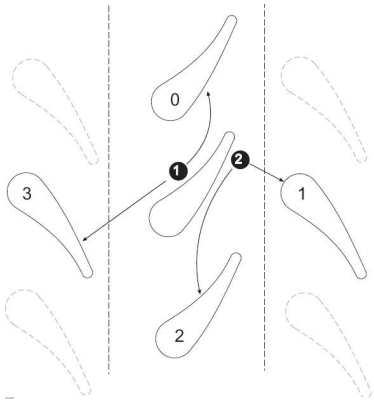

Figure 3: Close-up on the blade geometry. Robots can detect distinct points on the blade (① and ②) using their on-board sensors. These waypoints serve as "launch-points" to traverse to a neighboring blade. The arrows indicate pre-programmed trajectories.

the task. As tasks are allocated sequentially, single-item auctions suffer from potentially sub-optimal solutions if the order of task execution matters. In this case, optimal solutions can only be achieved when robots bid on all possible combinations of tasks. This concept is known as a *combinatorial* auction [3], but does not scale well with the number of tasks as the number of possible combinations grows exponentially.

The winning bid is commonly determined by a centralized entity that runs the auction and acts as auctioneer. If the rule for bid determination is deterministic, however, and all bids are known to each robot participating in the auction, the central auctioneer is not required as every robot can determine the winning bid itself [10].

In this paper, we tackle the multi-robot boundary coverage problem by trading tasks via single-item auctions using a deterministic rule for determinination of the winning bid. We chose single-item auctions for scalability reasons, as the number of bids does not only affect computation but also the communication load. Also, [20] provides provable lower bounds on the performance of the single-item auction.

Determining the task allocation a priori might lead to sub-optimal outcome when some of the robots do not execute the tasks allocated to them as desired. We therefore extend the algorithm described in [10,20] for re-allocating the remaining task set whenever a robot discovers that it is late. In order to tackle dynamic team sizes, we also describe an algorithm for determining when auctions can be cleared when the number of team members is changing. In the remainder of this section, we will first outline the assumptions on the individual robotic platform (Section 4.2) and then define a cost fuction that is suitable for local bid computation (Section 4.3) and is the basis for calculating the global objective function (Section 4.4). We then describe the market-based algorithm that is used for the on-line initial task allocation (Section 4.5) and describe how the robots sequentially process the tasks allocated to them (Section 4.5.3). We will then describe an extension of the algorithm which continuously re-auctions the remaining tasks thus taking into account possible navigation errors and delayed task executions of individual robots (Section 4.5.4).

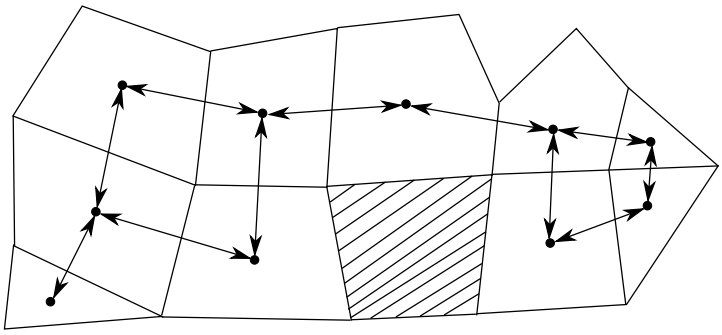

Figure 4: An arbitrary environment with a cellular decomposition $T \times E$ consisting of a set of tasks T (filled circles) and a set of edges E (arrows). Dashed cells are obstacles.

4.1 Formal definitions

More generally, we describe the cellular decomposition of the environment as a undirected, connected graph $G = (T, E)$ with m vertices T, called *tasks* and edges E, and n homogeneous robots defined by the set $R = \{r_1, r_2, \ldots, r_n\} \subseteq T$, that is a robot is represented by its position on the graph. Edges represent navigable routes between tasks and can be traversed by a robot in either direction. We denote $\tau \in T = \{\tau_1, \tau_2, \ldots, \tau_m\}$ an individual task, and $E^\tau = \{e_1^\tau, \ldots, e_{n_\tau}^\tau\} \subseteq E$ the set of n_τ edges incident to τ, so that the location of task τ can be reached from any other location by a finite number of edges.

We denote $\mathcal{A} = \{A_1, \ldots, A_n\}$ to be an *allocation* of T to the set R, where $A_i \subseteq T$ and tasks in A_i are allocated to robot r_i (with n the number of robots). We say the task allocation $\mathcal{A}$ to be complete when $\bigcup \mathcal{A} = \mathcal{T}$ (i.e. the task allocation is left-total, all tasks are assigned to at least one robot), and non-redundant when $\bigcap \mathcal{A} = \emptyset$ (i.e. the task allocation is injective, not more than one robot per task).

A sample environment with an arbitrary cellular decomposition is depicted in Figure 4. Our case study, in which all elements are aligned in a regularly spaced grid, is an instance of such an arbitrary graph and facilitates the implementation of the vertex-to-vertex navigation algorithms using our miniature platforms.

4.2 Assumptions on the robotic platform

We assume that the robots can determine all edges incident in τ and have a perfect knowledge about the structure of the graph (i.e. the number of vertices and navigable routes are known in advance). We also assume that robots are able to localize themselves within the cellular decomposition of the environment and communicate with their team-mates within a certain range.

4.3 Local Objective Function

Consider a non-negative cost function $\boldsymbol{J}_l(r, \tau)$ with $r, \tau \in R \cup T$, which represents the local cost for robot r to accomplish task τ given its initial conditions (e.g., given position in the environment). We assume the cost function to be the same for each robot and symmetric, i.e. $\boldsymbol{J}_l(r, \tau) = \boldsymbol{J}_l(\tau, r)$.

The cost function $\boldsymbol{J}_l$ can take different metrics into account, e.g. distance, time, battery

level, probability of robot failure, and might also suffer of significant noise. For the cost function $\boldsymbol{J}_l$ we chose the shortest distance between r and τ with an uniform edge weight of 1. Given $\boldsymbol{J}_l$, we can also calculate the cost of the *shortest path* for accomplishing all tasks of an allocation A_i by a robot r_i starting from its current position denoted by $SP(r_i, A_i)$. The cost of the shortest path is also referred to as *total cost* in the remainder of this paper. Computing the shortest path visiting a set of locations and visiting each of these only once is an instance of the Traveling Salesman Path Problem (TSPP)[4].

4.4 Global objective function

Depending on the application, the global cost function can have different emphasis, e.g. minimize the sum over all robots' path costs, or minimize the overall energy consumed. In our implementation, we focus on minimizing makespan, i.e. the total time needed for complete accomplishment of all tasks (i.e. all the boundaries of the objects of interest covered). This is expressed as

$$\boldsymbol{J}_g = \min_{\mathcal{A}} \max_{j} SP(r_j, A_j) \tag{1}$$

or in words, to find a task-allocation that minimizes the length of the longest path any of the robots has to travel.

4.5 Algorithm

Initially, all tasks are unallocated and robots are deployed in the environment (scattered or at a central location). As the initial position of the robots is unknown until they have reached the first blade, the initial position in their local information are initially set to *unknown*. An unknown position prevents the robot from taking part in auctions. As soon as a robot reaches a blade and determines its position, it broadcasts that it is ready to auction. As soon as all robots are ready (this can be established locally if the total number of robots is known or using a time-out otherwise), the auction for the initial task allocation begins.

At each round of the bidding process, each robot calculates bids on all unallocated tasks. The bids are calculated based on the cost a task would have given the robot's current position and tasks already assigned to it. Information about the number of unallocated tasks is kept locally and is based on local task progress as well as communication with other robots. Each robot broadcasts its bid with the lowest cost, and therefore having the highest chance of winning the auction, to the team. The robot which submitted the overall best bid for a given task — an information ideally known to all robots — will be assigned to it. New rounds take place until all tasks have been allocated. With m tasks, this represents m rounds. Notice that this paper is not concerned with algorithms for reliable transmission of the bids to the robot team, which is assumed to be taken care of by the wireless transmission protocol.

Upon complete allocation, each robot is responsible for a set of tasks, which are executed sequentially by following a shortest path connecting all tasks (see Section 4.3). As every robot only bids using its local objective function, the number of robots can exceed the number of unallocated tasks and therefore one or more robots can end up without tasks assigned at the end of the task-allocation process.

[4]The original Traveling Salesman Problem (TSP) requires the traveler to return to its initial position.

We will now define the deterministic rule that is used by each robot to determine the winning bid in an auction, and then finally formally define the task-allocation process.

4.5.1 Bid Evaluation

We will now define what makes a bid better than another. Bid evaluation is critical for tracking the desired global objective function as accurately as possible (see [27] for the generation of bidding rules as a function of the desired global objective function). A key requirement for the bid evaluation rule for decentralized execution is that the evaluation is unambiguous, i.e. for the same set of bids, two robots must elect the same best bid.

We define a bid b from the robot r_i on the unallocated task τ to be a 4-tuple $b = (i, \tau, tc, mc)$, where b^{tc} represents the bid's *total cost* and b^{mc} the bid's *marginal cost*. The total cost for a robot r_i bidding on a new task τ is given by the shortest path from its current position over the current partial allocation at auction round k, which we denote with $(S_i)_k$, and a new task τ, i.e.

$$b^{tc} = SP(r_i, (S_i)_k \cup \{\tau\}) \tag{2}$$

The marginal cost represents the difference between the total cost with and without the unallocated task the robot bids on, i.e.

$$b^{mc} = SP(r_i, (S_i)_k \cup \{\tau\}) - SP(r_i, S_i) \tag{3}$$

We now define whether a bid is better than another using a binary dominance relation Δ. A binary relation[5] between two sets is defined as a subset of the Cartesian product of the two sets, i.e. $\Delta \subseteq B \times B$ describes a binary relation between elements of the set B of all possible bids. We say that the ordered pair (b_j, b_k) with $b_j, b_k \in B$ meets the relation Δ if $(b_j, b_k) \in \Delta$ and write $b_j \Delta b_k$. Using this concept, we can define a binary relation that expresses a bid being better than (or dominating) another one.

Definition 1 ("better than" or "dominating" binary relation)**.** *Consider a bid $b = (i, \tau, tc, mc) \in B$. Two bids, $b_i, b_j \in B$ with $b_i \neq b_j$, satisfy the relation $b_i \Delta b_j$, if b_i "is better than" or "dominates" b_j according to the following rule:*

$$\begin{aligned} (b_i^{tc} < b_j^{tc}) \quad & \vee \\ (b_i^{tc} = b_j^{tc} \wedge b_i^{mc} < b_j^{mc}) \quad & \vee \\ (b_i^{tc} = b_j^{tc} \wedge b_i^{mc} = b_j^{mc} \wedge i < j) \quad & \Rightarrow \quad b_i \Delta b_j \end{aligned}$$

Δ is unambiguous as the robot IDs are unique by definition.

Notice that there can be two or more bids with the same total and marginal costs, we make use of the robot ID as a last resort for deterministic selection (with priority to small values).

This allows us to formulate the following bidding rule. Consider $(S_1, S_2, \ldots, S_n)$ to represent the partial allocation of tasks T in some round of the auction for the n robots and τ an unallocated task. A robot r_i will now bid on τ using the following bidding rule.

Definition 2 (Bidding rule)**.** *Robot r_i selects, among the unallocated tasks set, the task τ which minimizes the local objective function with respect to its current tasks allocation and broadcasts the corresponding bid $b = (i, \tau, tc, mc)$.*

[5]Commonly defined binary relations are operators such as $>$, $<$, $\leq$, etc.

4.5.2 Single-Auction Task-Allocation Process

We can now formally describe the task allocation that is achieved by a single auction. For brevity, the index i used on sets is used for indicating the association of this set with robot r_i. The variables used in the task-allocation process are summarized in Table 1.

Variable	Description
$A_i^t \subseteq T$	The assigned tasks set to robot r_i at time t.
$C_i^t \subseteq T$	The set of accomplished tasks by robot r_i at time t communicated to all the robots.
$(P_i)_k \subseteq T$	The unallocated task set used during the task-allocation process by robot r_i at round k of an auction.
$(S_i)_k$	A partial allocation of tasks at round k of an auction.
$(B_i)_k$	The set of all local bids for robot r_i at a given round k. B_i has the same cardinality as P_i and is recalculated at the beginning of each round.
$(b)_k = (i, \tau, tc, mc)$	The single bid broadcasted by robot r_i at each round.
B_i^*	The set of all broadcasted bids received by robot i at a given round.
$(b^*)_k = (j, \tau, tc, mc)$	The best overall bid at a given round. Task τ is allocated to robot r_j.

Table 1: Summary of variables used in the task-allocation process.

A robot r_i's state is described by the set of tasks $A_i^t \subseteq T$ assigned to it and the set of tasks $C_i^t \subseteq T, \quad \forall t \geq 0$ that a robot knows that have been successfully carried out (by itself or by other robots) at time t. During the auction process robots base their bids on the unallocated task set P_i^t, which is given by the coverage progress C_i^t at time t. The set of unallocated tasks at round k, $(P_i^t)_k$, is initialized with the set of all uncovered tasks at time t, corresponding to

$$(P_i^{t=0})_0 = T \tag{4}$$

for an auction being held at the beginning of the experiment. The set of unallocated tasks (P_i^t) does not change during an auction, and $(P_i^t)_0$ will be only evaluated once at the beginning of an auction. Thus, the index t is omitted in the context of an auction.

Compute the local bid At each round k of the auction, robot r_i computes the set $(B_i)_k$ of cardinality $|(P_i)_k|$ which contains the bids on all unallocated tasks. Robot r_i then selects locally its single bid b_k with

$$b_k = b | b \in (B_i)_k \wedge b \Delta b', \forall b' \in (B_i)_k \{b\} \tag{5}$$

and broadcasts it.

Elect locally the best overall bid Consequently, robot r_i receives the bids from the other robots, defined as the set $(B_i^*)_k$, and locally selects the best overall bid b_k^* for round k with

$$b_k^* = b | b \in (B_i^*)_k \wedge b \Delta b', \forall b' \in (B_i^*)_k \{b\} \tag{6}$$

as soon as it received bids from all robots.[6]

For global communication, at each round each robot receives a bid from all of the other robots, i.e.

$$(B_i^*)_k \quad = \quad (B_j^*)_k, i \neq j, \forall i, j \in R \tag{7}$$

Task assignment and auction clearance Let robot r_j be the robot that submitted the best overall bid according to (6) on the unallocated task τ at round k, it updates its task allocation S_j by

$$(S_j)_{k+1} = (S_j)_k \cup \{\tau\}, \qquad (S_j)_0 = \varnothing \tag{8}$$

while all robots $r_i \in R$ update the set of unallocated tasks by

$$(P_i)_{k+1} = (P_i)_k \setminus \{\tau\} \tag{9}$$

where $(P_i)_0$ has been initialized according to (4). The auction process continues until $(P_i)_k = \varnothing, \forall r_i \in R$.

At the end of task allocation, each task has been assigned to one robot responsible for inspecting it and a robot updates its set of tasks A_i^t by

$$A_i^{t'} = S_i \tag{10}$$

where t' reflects the time at which the last auction has been cleared.

4.5.3 Task Execution

As part of the task-allocation process, each robot computes a shortest path, covering all assigned tasks, which will be followed during coverage. To determine the path used to navigate from a task τ_1 to a task τ_2 ($\tau_1, \tau_2 \in T$), we calculate the shortest path between the location of τ_1 and τ_2 so as to minimize $\boldsymbol{J}_l$ (see Section 4.3). If there is more than one path that minimizes $\boldsymbol{J}_l$, the path containing the most incomplete tasks is chosen. Although this situation cannot arise in a deterministic, optimal scenario as by definition tasks are connected by the shortest path, it might be generated as an artifact of non-optimal solutions for the shortest path as well as of navigation errors of the robots.

After having carried out a task τ robot r_i will update its state with

$$A_i^{t+t^b} \quad = \quad A_i^t \setminus \{\tau\} \tag{11}$$

$$C_i^{t+t^b} \quad = \quad C_i^t \cup \{\tau\} \tag{12}$$

and then broadcast $C_i^{t+t^b}$. The time t^b corresponds to the time the task execution takes.

Notice that a robot shares its full completed task set, and not only the last completed task, in order to increase robustness towards unreliable communication and limited communication range. Thus, at any time during the coverage, robot r_i can receive information concerning inspected tasks, C_j^t, from any other robot $r_j \in R \setminus \{r_i\}$. It will update its own state by merging the two completed tasks set, that is

$$C_i^{t+t^e} = C_i^t \cup C_j^t \tag{13}$$

[6]The implicit assumption of knowing the number of robots in the team will be relaxed later in the paper.

where t^e is the necessary time for communication and information processing. Notice that in our scenario robots navigate always on a graph defined by task locations. When robots move from a given task location to that of a task to be completed, they might traverse several other task locations, perhaps previously assigned, and therefore the boundary of the corresponding object might already have been covered. The robots will in this case again cover the objects's boundaries and broadcast the tasks as accomplished once again.

4.5.4 Task Re-Allocation

The task allocation can only be executed when none of the robots fail and when perfect localization is available. Both of them are a strong assumptions on miniature robots since robots might run out of battery, get stuck, or physically break. Also, navigation errors will lead to sub-optimal coverage paths that might be improved by re-planning. If communication is neither global nor perfectly reliable, we will show that task allocation is still complete although non-redundancy of task allocation cannot be ensured any longer. In order to deal with the limitations of a real (miniature) robotic platform, we make use of a continuous re-auctioning process, which is started whenever a robot reaches a new task. For doing so, we have also to relax the assumption on *a priori* knowledge of the number of robots participating in an auction. Indeed, in a real system, not all robots might be within the communication range of each other or face communication failures and some robots might have failed or being replaced during task execution. Therefore we introduce a *time-out* as terminal condition for a bidding round instead of waiting until bids from all robots have arrived. Depending on the choice of the time-out length, smaller or larger sub-teams will emerge. As every sub-team will still need to allocate all tasks in the environment, each sub-team will calculate complete allocations with potential redundancy. Using time-outs is thus a trade-off between speed of the task allocation (or re-allocation) and redundancy. Coverage redundancy can thus be expressed as a function of the time-out length which should be designed as a function of the communication reliability and connectivity. This can be illustrated by considering two worst-case scenarios. In the first, the time-out is infinitesimally small (corresponding to no communication possible) and task execution is fully redundant. In the second, the time-out is infinitely long: in case of perfect communication, the allocation is quick and non-redundant; in case of unreliable or limited-range communication the team can incur in infinitely long deadlocks although in principle without generating coverage redundancy. In other words, the longer the time-outs, the lower is the coverage redundancy, and the slower is the response of the task-allocation to dynamic changes in the robotic team or the environment.

In our implementation, an adaptive time-out mechanism has been chosen. We select a small initial time-out value, which remains constant during the experiment. Whenever a new bid is received, the timer is reset and stretches the effective waiting time. This policy shows two advantages. First, it allows for fast auction closing as robots wait only a short time after the last bid has been received. Second, the effective waiting time scales with the actual size of the sub-team engaged in the same auction, but is independent of the number of robots expected to participate in the auction.

We see that with the potential for total robot failure, the set of robots $R(t)$ is time-dependent and the multi-robot task-allocation problem needs to be solved whenever the system changes in an unforeseen manner. For a discrete decomposition of the environment, the task-allocation problem potentially needs to be solved every time a robot update its position by moving from

one task to another. Thus, after reaching a task τ, the robot broadcasts its completed task set C_i^t, but also initiates an auction on the remaining tasks within its sub-team.

Unlike defined by (4), the set of unallocated tasks $(P_i^t)_0$ at the beginning of every new task-allocation process is a function of the current progress known to the sub-team robot r_i is part of.

$$(P_i^t)_0 = T \setminus C_i^t \tag{14}$$

The auction process is then executed using (8) and (9) until $(P_i)_k = \varnothing$.

5 Deterministic Analysis

We will now formally analyze discrete properties such as completeness and efficiency of the proposed algorithm. Efficiency is measured in terms of redundancy, i.e. whether tasks are executed more than once. We will first show that the single-item auction algorithm yields complete and non-redudant allocations (Lemma 5.1) if all robots can communicate with each other and show that coverage is eventually complete when all robots can recover from navigation errors using localization (Theorem 5.2). We then relax our assumptions on the communication requirements and allow for permanent failure of some of the robots (Theorem 5.4). Finally, we provide upper bounds on the required computation, communication, and memory.

5.1 Single-Item Auction without Task Re-Allocation and Fixed Team Size

Lemma 5.1 (Completeness and Non-Redundancy of Task-Allocation)**.** *At the end of the task-allocation process, each task in a perfectly communicating team has been assigned to one and only one robot. That is* $\bigcup_{i\in\{1\dots n\}} S_i = T$ *and* $\bigcap_{i\in\{1\dots n\}} S_i = \varnothing$.

Proof. As all robots can communicate with each other in a communicating team and as Δ is unambiguous as the robot IDs are unique, the allocation is not redundant. All robots agree on the same overall best bid for each round, and remove it from the set of unallocated tasks. As auctioning takes place until all tasks are allocated, the allocation is complete. □

Lemma 5.1 relies on perfect communication; see Section 4.5.4 for a detailed discussion when this assumption does not hold. Notice that non-redundancy of task-allocation is not equivalent to coverage non-redundancy. In fact, as robots navigate between tasks on the shortest possible route they will visit previously covered regions, which leads to redundant coverage in this case study. This is in particular the case, if the scenario contains bottlenecks where the robots must pass through in order to reach a certain area. In applications where coverage and navigation policies are implemented separately, e.g. painting or inspection, coverage can be made non-redundant if task-allocation is non-redundant.

Theorem 5.2 (Complete Coverage of the Environment)**.** *Given that the task allocation is complete, coverage is complete if none of the robots fail permanently and every robot can diagnose and recover from an error using localization. At the latest, complete coverage is achieved when all robots have executed all of their tasks. In any case, coverage is complete when* $\bigcup_{i\in R} C_i^t = T$.

Proof. If no robots fail permanently, all robots will eventually complete all tasks that have been assigned to them as the assumption of perfect localization allows them to recover from navigation errors. As the task-allocation is complete (Lemma 5.1), coverage is complete. As

navigation errors might lead to sub-optimal coverage paths, coverage might already be complete before each robot has explicitly terminated its partitioning of the environment. □

5.2 Single-Item Auctions with Task Re-Allocation and Variable Team-Size

We will now show that even when relaxing the assumptions on robot failure and perfect communication, task re-allocation allows for maintaing provably completeness as long as at least one robot recovers from failure in a finite time. For this, we will first show that the algorithm is dead-lock free, i.e. a task assigned to a robot that will fail before executing this task can still be assigned to another robot.

Lemma 5.3. *Upon robot failure, a task previously assigned to this robot will be eventually executed if at least one robot does not fail.*

Proof. By definition, robots broadcast task completion only after actually executing a task. If a robot fails before broadcasting task completion or if it is out of communication range of any other robot sub-team, it will consequently not take part in the next task-allocation process of this sub-team. As the task on which the robot failed is known as being incompleted by the sub-team, this task will be assigned at least to one other robot. □

Theorem 5.4 (Complete Coverage without Perfect Communication)**.** *Coverage is complete if at least one robot recovers from failure in finite time and localization information is perfect.*

Proof. Under the absence of perfect communication (e.g., limited range, unconnected network, unreliable) the robot team is split into fully connected sub-teams of at least one robot. Task-allocation within each of these sub-teams is complete and non-redundant (proof follows directly from Lemma 5.1). Each of the sub-teams will achieve complete coverage as the algorithm is dead-lock free (Lemma 5.3). Thus, if at least one robot does not fail, a sub-team will exist that provide complete coverage. □

5.3 Algorithm Complexity

We study complexity of the algorithm with respect to computation time, memory, and communication requirements at the level of the individual robotic node. First, we discuss the complexity of a single auction, and second, we extend the complexity to the algorithm with re-auctioning.

5.3.1 Single-Item Auction Complexity

During each round, the robot needs to first compute the bid to be broadcasted. This represents for robot r_i and m tasks the calculation of one shortest path including the previously assigned tasks for each unallocated tasks. Given $\mathcal{O}(m^2)$ the complexity of an insertion heuristic used for solving the TSPP [2] and $\mathcal{O}(m^2)$ the complexity of Dijkstra's Shortest Path algorithm, which needs to be executed once per auction, this leads to a computation complexity of $\mathcal{O}(m^2)$. As one task is assigned at each round, the computation complexity for one auction is $\mathcal{O}(m^3)$. This is an upper bound as the number of unallocated tasks decreases during the coverage. At the end of each of the m rounds, the best overall bid is elected over the n total bids, leading to an overall computation complexity of

$$\mathcal{O}(m^3 + nm) \tag{15}$$

Each robot broadcasts one bid per round and receives potentially a bid from each $n-1$ other robots. As robots do not pass on messages, the *communication complexity* is $\mathcal{O}(nm)$.

If higher bounds on optimality are required than provided by an insertion heuristic for solving the TSPP, the computational complexity needs to be adjusted accordingly.

5.3.2 Single-Item Auction with Task-Reallocation

In absence of navigation errors, the re-auctioning algorithm increases the auction complexity by a factor proportional to m, since each task potentially represents a new auction, leading to the complexity $\mathcal{O}(m^4 + nm^2)$ and $\mathcal{O}(nm^2)$ for communication requirements.

For high levels of noise and resulting navigation error, however, task completion might take infinite time, and thus yield to an infinite number of re-auctioning processes. In practice, coverage will eventually complete and the number of re-auctioning processes will be cm, with c being a constant factor and $c \ll m$. Notice that the worst case in computational complexity happens when no communication is available. Each robot will potentially run one auction for each reached blade as the task-allocation process results in a fully redundant task accomplishment. This implies the computation complexity to be increased by a factor proportional to n.

The *memory* requirements are $\mathcal{O}(m)$ as each robot needs to store information on all tasks.

6 Microscopic Modeling and Simulation

Whereas deterministic analysis (Section 5) allows for investigating completeness and efficiency, the probabilistic nature of the indivdual robot behavior (given by sensor and actuator noise), makes analysis of quantitative properties such as time to completion (Section 4.4) difficult. We thus investigate the performance of the algorithm at two different abstraction levels. Both models are microscopic, i.e. characterized by individual representations of each robot, and are exact representations of the algorithm described in Section 4.5. Both models differ in the way the interactions of the robots with the environment are modeled. Parameters of the algorithm such as t^b (for instance in Equation 11) correspond to measured values in the less abstract model, which carefully simulate a realistic instance of the robot team (taking into account the robot's body shape and sensorial configuration), whereas they are considered random variables in the more abstract model.

In both models robots navigate to the next vertex that is closest to the next task and update their allocation and coverage map using Equations 11–13. In case an auction takes place, a robot calculats bids on the remaining tasks using Equations 2–3 and 5–7 until all tasks are allocated using Equations 8 and 9, and finally determine update its task allocation using Equation 10. For the first auction, the set of unallocated bids is initialized using Equations 4 and Equation 14. We use Dijkstra's algorithm for calculating the shortest path and an insertion heuristic with provable lower bound [2] for solving the TSPP.

6.1 Realistic, module-based simulation

At the lowest abstraction level, we simulate the environment and the robotic platform in the realistic, module-based simulator *Webots* [24] (see Figure 1, *right*). Webots is able to faithfully reproduce detailed features of discrete, intra-robot modules (e.g. sensors, actuators, and transceivers), including their non-linearities and noise (for instance, in our simulations, different

levels of wheel slip were tested) and can accurately simulate the sensor and actuator configuration of the Alice robot. We implemented the robot controller described in Section 3.2 exactly in the same way as on the real Alice. Consequently, navigation on the graph is achieved by faithfully simulating sensor-based navigation in the blade environment, whereas communication with other robots is limited by a maximum communication range, specified by the radius of an imaginary disc with a robot at its center.

6.2 Agent-Based Discrete Event System Simulator

We also implemented an agent-based DES simulator in *JAVA*. Unlike Webots, the DES simulator models robots as bodyless agents moving from vertex to vertex on the graph representing the environment. Multiple agents can occupy one vertex, without interfering with each other. Instead of simulating the vertex-to-vertex navigation realistically, we calculated the histogram of the time t^b needed for covering one vertex (including edge traversal) for 6000 instances in Webots with one robot (i.e. no simulated physical interference with other robots possible). Results are reported in Figure 5. Out of this distribution we randomly sampled the time one DES simulation step is taking for one robot. The time needed to reach the first blade at the beginning of the experiment is also based on these values. Hence, results from the DES simulator are provided in seconds (rather than number of elements covered) and can be compared with the results of Webots simulation and real robot experiments. In the DES simulator communication range is specified in terms of neighborhood relations on the graph: same vertex, direct neighborhood (one edge), or global.

While Webots simulates wheel slip by adding to the individual desired wheel speed a variable amount of noise drawn from a constant, uniform distribution, navigation errors in the DES simulator correspond to a probability of failing when moving from vertex to vertex. In case of failure, a robot is randomly positioned on a vertex in the neighborhood (one of the four neighboring blades at one edge distance) of its current location. Table 2 summarizes probabilities to successfully navigate from blade to blade for different simulated and real wheel slip values for the Alice robot (π^e), the average number of inspected blades before failure (μ), and the time t^b required for covering one blade [6].

Wheel slip	π^e	μ	$t^b[s]$
0%	1	25	12
10%	0.79	4.77	38.5±10.3
50%	0.67	3.03	44±20.5
real	0.64	2.79	52±19.7

Table 2: Calibrated model parameters for different amounts of wheel slip in realistic module-based simulation and real robots. Different amounts of wheel-slip in simulation (numeric values) or measured on a real platform are translated into a probability π^e of successfully traversing an edge, the average time before failing μ, and the average time required for navigating from vertex to vertex t^b.

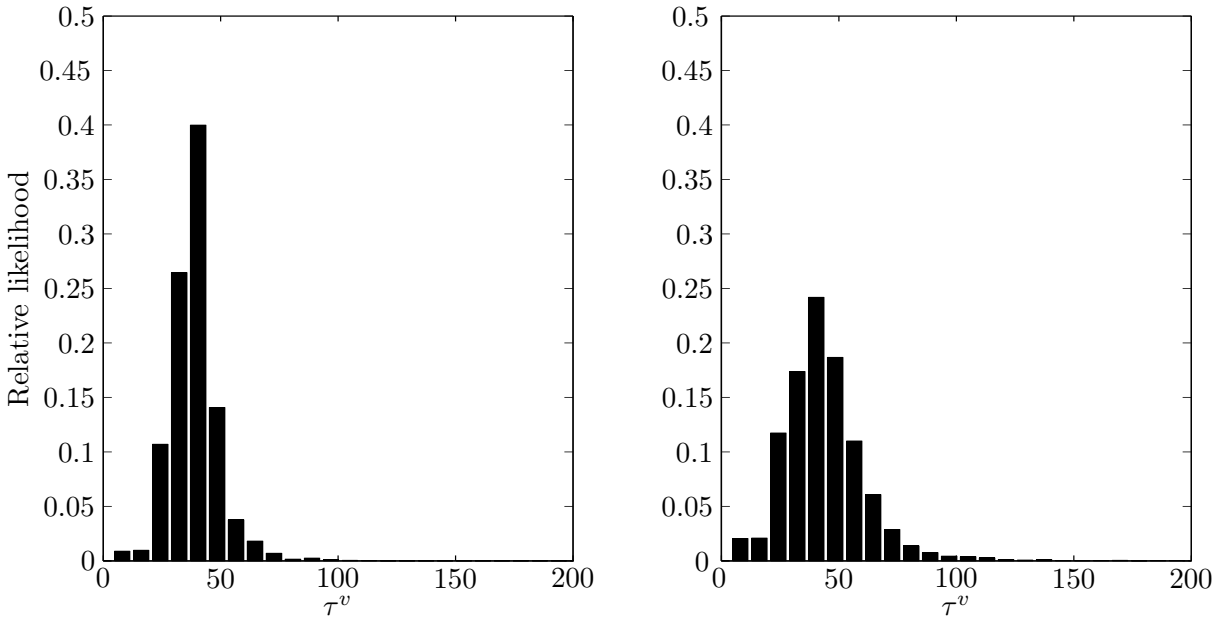

Figure 5: Histogram of possible durations (in seconds) for executing one task resulting from 6000 trials using Webots for wheel slip of 10% (*left*) and 50% (*right*).

7 Results

We conducted 100 simulations for each configuration in Webots or DES simulation. As coverage performance shows a long-tail distribution, we provide the median time to completion and its 95% confidence interval for each experiment (error bars). Unless otherwise specified, robots are spread randomly within the environment at the beginning of each experiment. Also, all experiments are conducted in the 5x5 grid environment corresponding to 25 blades to be inspected (i.e. 25 tasks to be allocated). On all abstraction levels, robots are subject to sensor and actuator noise, be it due to their physical implementation, realistic simulation, or probabilistic simulation based on calibrated values. This might lead to very long task execution times, which prevent robots from recovering before the end of the experiment and could be described by a complete failure of this robot. The parameter t^e (communication processing time) has been set to zero in all our experiments.

7.1 Consistency for Different Implementation Levels

For calibrating the DES simulation, we recorded the necessary times for covering 6000 blades in Webots using wheel slip of 10% and 50% (see Figure 5) with a faithfully implemented model of the Alice robot (see [6] for the calibration experiments of the Webots implementation). Drawing task execution times randomly from these distributions, predictions of the DES simulator and Webots for wheel slip of 10% and 50% provide close match and results suggest that the performance gracefully decays with increasing amount of slip noise (see Figure 6). This allows us to explore other properties of the algorithm by relying on the computationally cheaper DES simulation (around two orders of magnitude difference on the same machine). We explored real-robot experiments using 5 robots (9 experiments) endowed with global communication and the re-auctioning algorithm (see Figure 7), which are well inline with DES and Webots simulations.

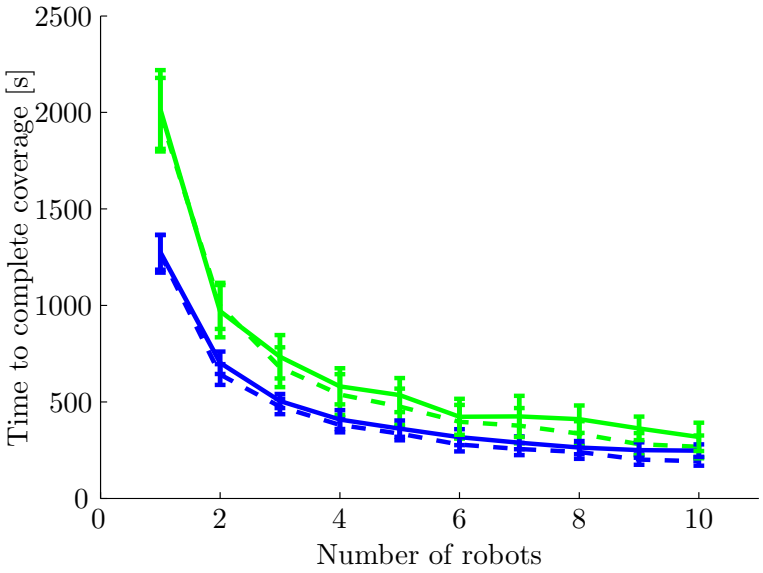

Figure 6: Prediction of the DES simulator (– –) and Webots (—) for wheel slip of 50% (upper two curves) and 10% (lower two curves) using a market-based coordination approach with re-auctioning provide close agreement and show that performance gracefully decays with increasing amount of slip noise.

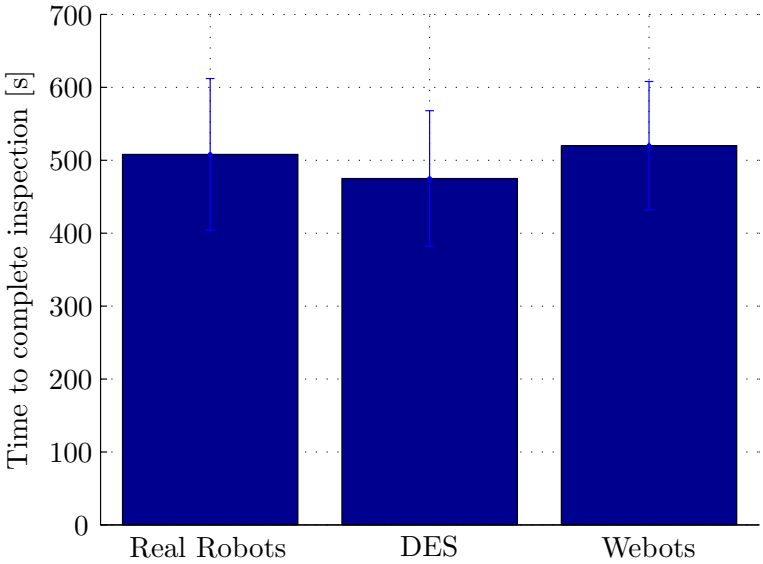

Figure 7: Time needed to achieve complete coverage for 5 real robots, Webots and DES simulation (50% slip-noise) with re-auctioning and global communication.

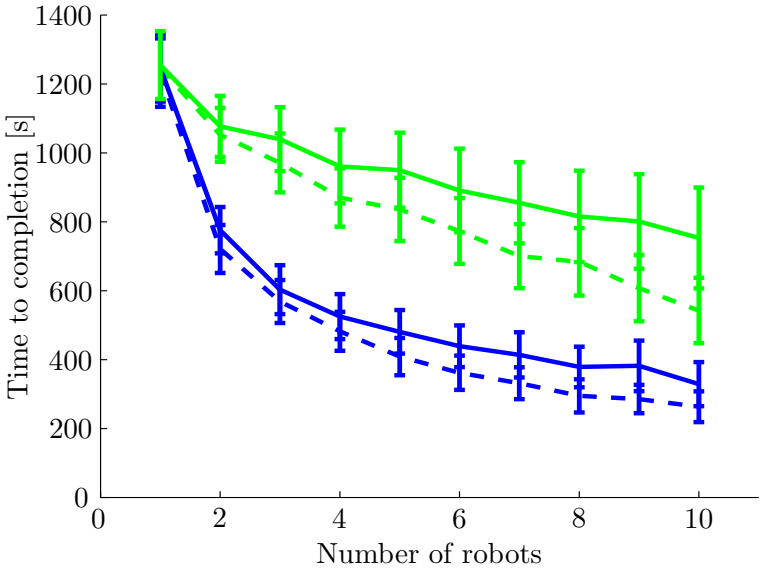

Figure 8: Time to completion with low wheel slip (10%) and without re-auctioning mechanism. We can compare DES simulation (– –) and Webots (—) with (lower two curves) and without communication (upper two curves).

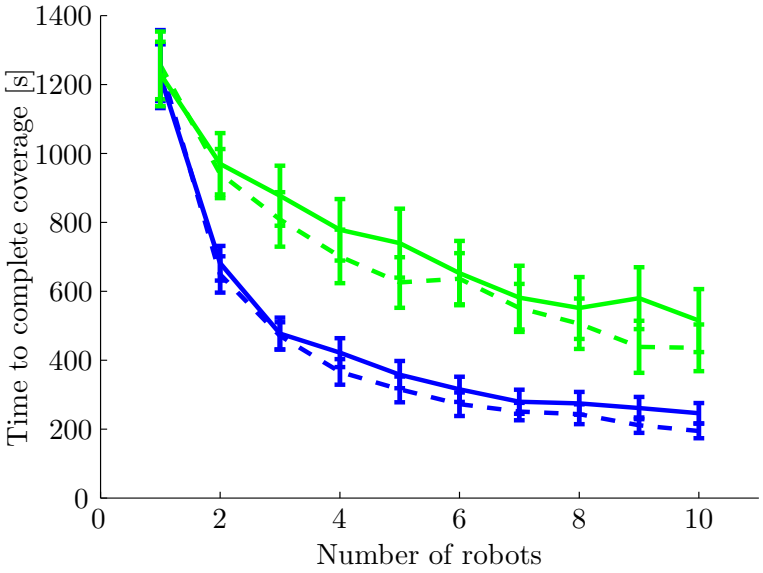

Figure 9: Time to completion with low wheel slip (10%) and re-auctioning mechanism. Prediction of the DES (– –) simulator and Webots (—) for no communication (upper curves) and global communication (lower curves).

7.2 Communication Range

DES and Webots are also used to study the impact of the communication range. Figures 8 and 9 show results for global communication and no communication (zero range), with and without re-auctioning.

Figure 8 depicts time to completion for DES simulation and Webots considering global communication and no communication, both without re-auctioning. Notice that when robots do not communicate, they do not need to complete the entire mission but only those parts that are allocated to them before deployment. As robots carry out coverage also when navigating in the environment, communication helps reducing coverage time even without re-auctioning. In Figure 9 we observe the beneficial effect of re-auctioning, i.e. performance is better with and without communication. When robots do not communicate, re-auctioning will lead to re-planning on individual level, which allows the robot to continue after navigation error without strictly enforcing the path it has been initially planned. With communication re-auctioning will lead to re-planning at collective level.

We observe a more important difference between DES and Webots simulation values without communication than with communication. When re-auctioning is used (see Figure 9), the difference is less important for both communication ranges. These are results of the different approximation introduced at both modeling levels which we will comment in details in Section 8.2.

As global communication is a strong assumption that can not always be guaranteed, we investigated using the DES and the Webots simulators four communication ranges: no communication (range 0), robots communicate only when they are on the same vertex (range 1), robots communicate only with robots on a neighboring vertex (range 2), and global communication (infinite range). For all communication ranges we consider reliable communication. All simulations are performed with a fixed wheel slip (10%). Figure 10 shows a comparison of the performance for these different communication ranges with re-auctioning, using Webots. We see that even a very limited range (i.e. same vertex) enables good performances, and in the 5x5 environment neighborhood communication comes close to the performance of global communication. This is not only due to the fact that a robot with communication range 2 covers 9/25 of the overall area (in the best case), but also because information propagates through the environment when the robots move.

7.3 Wheel Slip

As wheel slip of an individual robot can affect significantly the performance of the team, we investigate the impact of different levels of wheel slip on robots endowed with global communication. Wheel slip acts on the performance not only by being responsible for navigation errors but also by delaying task execution. When a robot gets behind schedule re-auctioning might help to distribute its remaining tasks among the team. The simulation are performed using the DES simulator. In Figure 11, we observe that re-auctioning helps to deal with significant wheel slip. Figure 11 also shows that the re-auctioning mechanism is able to partially or completely compensate the effects of wheel slip, especially for large teams of robots (8 and higher).

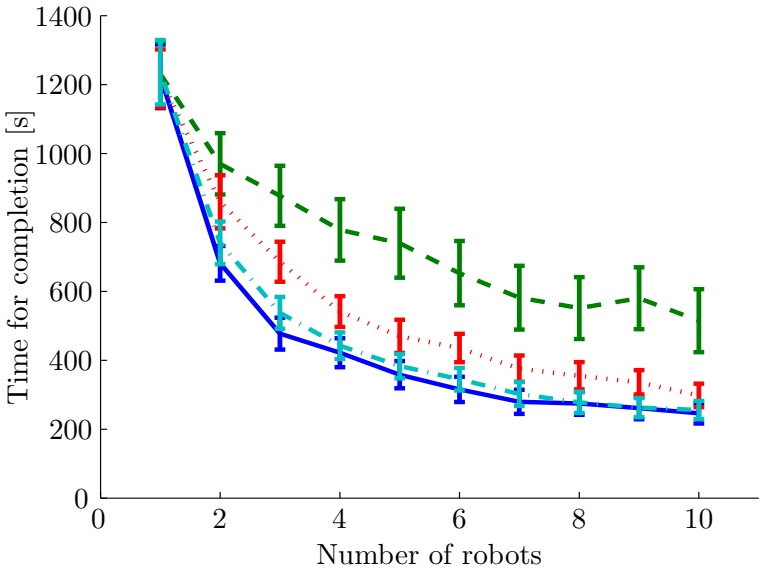

Figure 10: Time to completion with low wheel slip (10%) using Webots for no communication (– –), same vertex communication (· · ·), neighborhood communication (–·–), and global communication (—) with re-auctioning.

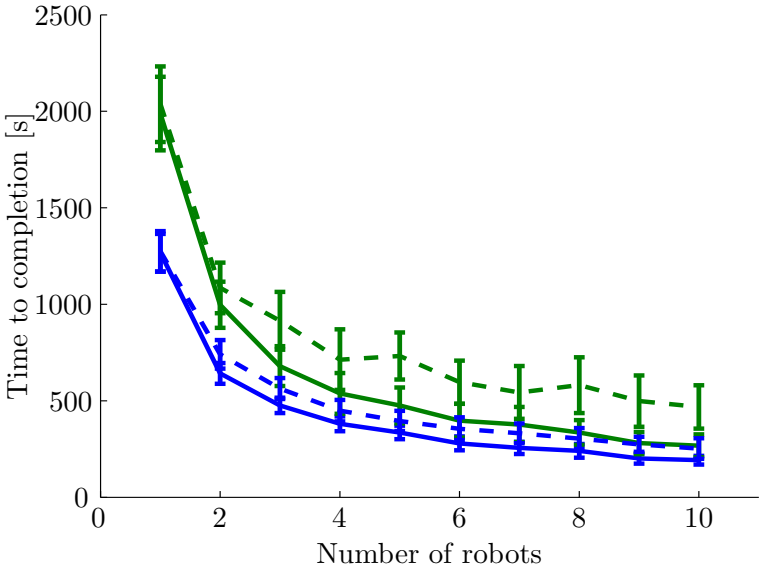

Figure 11: Time to completion with global communication using the DES simulator for 50% (upper two curves) and 10% (lower two curves) of wheel slip without (– –) and with (—) re-auctioning. Re-auctioning is increasingly beneficial for larger amounts of noise.

8 Discussion

Experiments using real robots highlight an important aspect of any multi-robot coordination algorithm: it might be useless to spend resources and time to compute an optimal solution, if this optimal solution is subject to constant degradation due to noise. Single-item market-based approaches together with heuristics for the individual robot cost function are a good compromise between solution quality and computation speed. Furthermore, single-item auctions are light in communication requirements (as opposed to combinatorial auctions, where the number of bids grows exponentially with the number of tasks) and offer the potential for distributing the computational load over the team as bid computation can be executed locally if the capabilities of the platform in use allows.

8.1 Computational Scalability of the Algorithm with Respect to the Environmental Size

The complexity of the insertion heuristic depends on the number of assigned tasks per robot. Moreover, the auction process also increases its length as a function of the number of tasks (m rounds needed to allocate m tasks). Therefore, in its current implementation, our algorithm does not scale well with the size of the problem, i.e. the number of tasks to be allocated, which is in turn determined by the size of the environment for a regular structure such as that of our inspection case study. The algorithmic variant with re-auctioning is suffering even more from this scalability issue. A potential solution to this problem could be to trade bundles of task, for example strips of the environment, instead of individual tasks. However, bundling tasks is a trade-off between individual computational complexity and quality of the solution.

8.2 Agent-Based vs. Module-Based Modeling

Both modeling abstraction levels provide generally good matching with real robot experiments (Figure 7). Although agent-based models (DES simulator) with only one robot provide essentially perfect alignment with module-based models (Webots simulator), we observe increasing differences between both modeling levels when multiple robots are involved. The cause for this divergence is that the module-based model accurately simulates physical interference between robots, which are neglected in the agent-based model. Indeed, the inspection time distribution (Figure 5) fed into the DES simulator does not take into account physical interference as only one robot was involved in the measurements performed with the Webots simulator.

By comparing upper and lower curves from Figures 8 and 9, we observe the highest divergence between both modeling abstraction levels when robots do not communicate. Without communication, all robots plan a shortest path covering all tasks and follow this path until completion. No planning coordination with the other robots is taking place, which potentially leads to additional physical interferences among the robots and therefore increasing discrepancies between the two modeling levels. Moreover, using re-auctioning in combination with communication further increases robot coordination and therefore minimizes the likelihood of physical interference, leading to an even better agreement between module-based and agent-based models (see Figure 9 lower curves vs. Figure 8 lower curves).

Thus, model predictions not explicitly taking into account physical interferences need to be treated with care for environments with a high density of robots or when robots do not explicitly

collaborate.

8.3 Benefit of Re-auctioning

As mentioned above, the divergence between DES and Webots simulations decreases with the use of re-auctioning. Another way to interpret this result is to say that physical interference, taken into account in the Webots simulator and managed by the robots reactively, is an additional source of noise on which re-auctioning can act in order to improve the solution. Also, our experience suggests that when adding noise to the evaluation of the local cost function, re-auctioning helps to decrease the effect of erroneous bids on the team performance. As noise on the local objective function can also be understood as an artifact of a potentially sub-optimal method for solution construction, this finding also supports our claim concerning a trade-off between solution quality and effective performance in environments subject to constant changes.

If the robot is drawn aside from its planned trajectory due to some external interaction (e.g., physical interference) or local noise (e.g., wheel slip), the robot will stick strictly to the initial plan if no re-auctioning takes place. In this case, a robot wastes significant efforts (time) and accumulates significant delays to recover a given position. This is translated at the team level by an expensive cost in terms of time, while not improving the global objective function. Figure 11 shows that this cost is proportional to wheel slip whereas the improvement of re-auctioning for higher wheel slip is more significant than that for lower wheel slip. In other words, the task re-auctioning efficiency is non-linear with respect to the noise level. Figure 11 clearly shows the benefit of re-auctioning, i.e. the performance with real wheel slip value can be brought back to the same level of performance as the one with only 10% wheel-slip but without re-auctioning.

8.4 Communication range

Figures 9, 10, and 11 show that global communication with re-auctioning offers the best performance. We will now discuss performance with respect to different communication ranges and re-auctioning.

8.4.1 No communication

If no communication is allowed, the initial partitioning does not result in distribution of tasks among the robots. Instead, each robot will win all of its bids and compute a global shortest path on its own without taking into account the contributions of the other robots, i.e. there is no explicit coordination.

Although no task partitioning takes place, we observe that re-auctioning without communication can significantly improve the performance. Re-auctioning without communication results in *local path re-planning*. The task partitioning does not change, but, by re-auctioning all tasks to itself, the robot recomputes a new shortest path covering all incompleted tasks. This helps to deal with noise (e.g., physical interference, wheel slip). On the other hand, without re-auctioning there is no local re-planning taking place and coverage will be highly redundant under influence of noise.

8.4.2 Limited communication range

When the communication range is limited, the quality of the task-allocation highly depends on the position of the robots during the auction process. For higher densities of robots in the environment, the probability to see robots within range of each other increases. In these cases, the use of local and global communication show similar improvement (see Figure 10 for big team sizes).

Limited communication range is also beneficial when considering the following special case: if all robots start at the same position, all robots will be able to participate at the initial task-allocation process, potentially leading to an optimal partitioning. When the robots get out of range from each other later on, the spatial distribution of the robots due to the initial task partitioning will minimize physical interferences between robots resulting in a more efficient and faster completion of the coverage mission.

8.4.3 Global communication

For global communication, all robots participate to the initial task-allocation process and further re-auctioning steps. This leads to the best overall solution in our case study. With re-auctioning, all robots will see dynamic task-allocation all along the inspection, which guarantees a shortest path each time they reach a new blade. This also allows them to have an accurate representation of coverage progress at any time, and thus minimizes coverage redundancy. Without re-auctioning global communication still yields an accurate initial task distribution, but the robot is forced to execute all tasks initially assigned to it despite navigation errors.

8.5 Influence of the initial distribution of the robots

We also investigated the effect of the initial robot distribution in the environment by comparing randomized vs. centralized deployments with and without re-auctioning and for different communication ranges. Whereas the performance for centralized deployments is generally lower than that of randomized ones due to navigation overhead generated by the crowded starting location, the algorithmic properties discussed above have shown to be persistent.

9 Conclusion

We described an algorithm for robust distributed multi-robot coverage of known environments by a team of robots. The robots perform task-allocation on-line using a market-based approach in a fully distributed way. The target system was modeled at multiple microscopic abstraction levels (using an agent-based, discrete-event system simulator and a module-based, realistic simulator) and implemented on a team of miniature robots *Alice*. All levels of implementation show close quantitative and qualitative agreement.

The microscopic models are then used to explore various noise levels and communication ranges including those measured on the real robotic platform. Using these results, we show that on-line re-auctioning of tasks helps to tackle sensor and actuator noise, not limited to robot failure but also including navigation errors which lead to delayed task execution and communication loss. We also show that even a very limited communication range offers significant improvement over no communication at all. The performance of the proposed algorithm scales

almost linearly with the number of robots when using communication. Thus, doubling the number of robots cuts the inspection time nearly in half, on the average, which is not the case when coverage is performed independently. Finally, results suggest gracefully decaying of the algorithm's performance for both decreasing navigation reliability and communication range.

In case of already moderatly unreliable robots, we observe a potential trade-off between computational efficiency and optimality of the obtained solution for the market-based task allocation algorithm. When the probability to successfully navigate is low, it is unlikely that a robot is able to complete all tasks allocated to it and it is likely that task re-allocation is required. In this case it might make sense to interrupt the task allocation process until a desired, potentially feasible, number of tasks has been allocated to each robot.

Although lower bounds on the performance of the proposed algorithms might be constructed by using the known lower bounds of the market-based task allocation algorithm and the lower bounds for the local cost function for specific scenarios (for example no robots ever fail, all but one robot fails, etc.), we showed that the effective performance of the algorithm is probabilistic and a function of sensor and actuator noise of the individual robotic platform. Thus, we provide probabilistic bounds on the performance in terms of confidence intervals obtained by DES simulation.

Repeated experiments with a team of 5 miniature robots show the viability of the approach on a platform with a high level of actuator noise (wheel slip) and a high rate of communication packet loss.

Acknowledgements

For performing this work, N. Correll and A. Martinoli have benefited of grants from the Swiss National Science Foundation (grant numbers PP002-68647 and PP002-116913).The authors would like to thank Xavier Raemy and Samuel Rutishauser for their support on hardware and firmware development, respectively.

References

[1] E. Acar, H. Choset, Y. Zhang, and M. Schervish. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *Int. J. of Robotics Research*, 22(7–8):441–466, 2003.

[2] V. Bafna, B. Kalyanasundaram, and K. Pruhs. Not all insertion methods yield constant approximate tours in the Euclidean plane. *Theoretical Computer Science*, 125(2):345–353, 1994.

[3] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt. Robot exploration with combinatorial auctions. In *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1957–1962, 2003.

[4] G. Caprari and R. Siegwart. Mobile micro-robots ready to use: Alice. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3295–3300, Edmonton, Alberta, Canada, August 2005.

[5] N. Correll. *Coordination Schemes for Distributed Boundary Coverage with a Swarm of Miniature Robots: Synthesis, Analysis and Experimental Validation*. PhD thesis, Number 3919, École Polytechnique Fédérale Lausanne, October, 2007.

[6] N. Correll and A. Martinoli. Robust distributed coverage using a swarm of miniature robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 379–384, Rome, Italy, April 2007.

[7] N. Correll, S. Rutishauser, and A. Martinoli. Comparing coordination schemes for miniature robotic swarms: A case study in boundary coverage of regular structures. In *Proc. of the Int. Symp. on Experimental Robotics (ISER)*, pages 471–480, Rio de Janeiro, Brazil, July 2006. Springer Tracts on Advanced Robotics, vol. 39, 2008.

[8] N. Correll, G. Sempo, Y. Lopez de Meneses, J. Halloy, J.-L. Deneubourg, and A. Martinoli. SwisTrack: A tracking tool for multi-unit robotic and biological research. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2185–2191, Beijing, China, Oct. 2006.

[9] M. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: a survey and analysis. *Proc. of the IEEE*, 94(7):1257–1270, 2006. Special Issue on Multi-Robot Systems.

[10] M. B. Dias and A. Stentz. A free market architecture for distributed control of a multi-robot system. In *Proceedings of the sixth Int. Conf. on Intelligent Autonomous Systems*, pages 115–122, Venice, Italy, July 2000.

[11] K. Easton and J. Burdick. A coverage algorithm for multi-robot boundary inspection. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 727–734, Barcelona, Spain, April 2005.

[12] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Min-max tree covers of graphs. *Operations Research Letters*, 32:309–315, 2004.

[13] G. Frederickson, M. Hecht, and C. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

[14] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31(1–4):77–98, 2001.

[15] B. Gerkey and M. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int. J. of Robotics Research*, 23(9):939–954, 2004.

[16] Brian P. Gerkey and Maja J Mataric. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation, Special Issue on Multi-robot Systems*, 18(5):758–768, October 2002.

[17] N. Hazon, F. Mieli, and G. Kaminka. Towards robust on-line multi-robot coverage. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1710–1715, Orlando, FL, USA, May 2006.

[18] M. Jäger and B. Nebel. Dynamic decentralized area partitioning for cooperating cleaning robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3577–3582, Washington, DC, USA, 2002.

[19] N. Kalra, D. Ferguson, and A. Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1170–1177, April 2005.

[20] M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, S. Koenig, A. Kleywegt, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, Cambridge, MA, USA, 2005.

[21] K. Lerman, C. Jones, A. Galstyan, and M. Matarić. Analysis of dynamic task allocation in multi-robot systems. *Int. J. of Robotics Research*, 25(4):225–242, 2006.

[22] K. Lerman, A. Martinoli, and A. Galystan. A review of probabilistic macroscopic models for swarm robotic systems. In *Proc. of the SAB 2004 Workshop on Swarm Robotics*, pages 143–152, Santa Monica, CA, USA, 2005. Lecture Notes in Computer Science Vol. 3342, Springer-Verlag, Berlin.

[23] A. Martinoli, K. Easton, and W. Agassounon. Modeling of swarm robotic systems: A case study in collaborative distributed manipulation. *Int. J. of Robotics Research*, 23(4):415–436, 2004.

[24] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[25] I. Rekleitis, A. New, and H. Choset. Distributed coverage of unknown/unstructured environments by mobile sensor networks. In Alan C. Schultz, Lynne E. Parker, and Frank Schneider, editors, *3rd International NRL Workshop on Multi-Robot Systems*, pages 145–155, Washington, D.C., March 2005. Kluwer.

[26] S. Rutishauser, N. Correll, and A. Martinoli. Collaborative coverage using a swarm of networked miniature robots. *Robotics & Autonomous Systems*, 2009. doi:10.1016/j.robot.2008.10.023 Available online 24 November 2008.

[27] C. Tovey, M. Lagoudakis, S. Jain, and S. Koenig. The generation of bidding rules for auction-based robot coordination. In *Multi-Robot Systems: From Swarms to Intelligent Automata*, volume 3, pages 3–14, 2005.

[28] K. Williams and J. Burdick. Multi-robot boundary coverage with plan revision. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1716–1723, Orlando, FL, USA, 2006.

[29] X. Zheng, S. Jain, S. Koenig, and D. Kempe. Multi-robot forest coverage. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3852–3857, Edmonton, Alberta, Canada, August 2005.

[30] R. Zlot and A. Stentz. Market-based multirobot coordination for complex tasks. *Int. J. of Robotics Research*, 25(1):73–102, 2006. Special Issue on the 5th International Conference on Field and Service Robotics.

[31] R. Zlot, A. Stentz, M. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3016–3023, Washington, DC, USA, May 2002.